AFRRI TR81-3

AFRRI TECHNICAL REPORT

Neurobiological data collection and initial data reduction system

D. O. Norman

DEFENSE NUCLEAR AGENCY
ARMED FORCES RADIOBIOLOGY RESEARCH INSTITUTE
BETHESDA, MARYLAND 20014

REVIEWED AND APPROVED

L Michael Fraser

L. MICHAEL FRASER
LCDR, MSC, USN
Chairman, Computer Sciences Department

Paul E Tyler

PAUL E. TYLER, M.D.
CAPT, MC, USN
Director

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFRRI TR81-3	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) NEUROBIOLOGICAL DATA COLLECTION AND INITIAL DATA REDUCTION SYSTEM		5. TYPE OF REPORT & PERIOD COVERED 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) D. O. Norman		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Armed Forces Radiobiology Research Institute (AFRRI) Defense Nuclear Agency Bethesda, Maryland 20814		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS NWED QAXM MJ 97410
11. CONTROLLING OFFICE NAME AND ADDRESS Director Defense Nuclear Agency (DNA) Washington, DC 20305		12. REPORT DATE September 1981 13. NUMBER OF PAGES 40
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number)		

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This system allows the collection and display of neurophysiological data during the time the experiment is being run. The system was initially designed for collecting both intracellular and extracellular evoked-response information. A 25-ms sample from two channels of analog information is taken periodically and processed. The two channels of information are sampled as closely to simultaneously as the instruction timing will allow (using an LSI-11, 30 μs). The operating mode and intersample period may be controlled

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. ABSTRACT (continued)

by the computer through a user-supplied time interval. It also may be driven by a Schmitt trigger through timing pulses during an experiment or from a recording.

CONTENTS

SECTION 1. OVERVIEW

This system allows the collection and display of neurophysiological data during the time the experiment is being run. The system was initially designed for collecting both intracellular and extracellular evoked-response information. A 25-ms sample from two channels of analog information is taken periodically and processed. The two channels of information are sampled as closely to simultaneously as the instruction timing will allow (using an LSI-11, 30 µs). The operating mode and intersample period may be controlled by the computer through a user-supplied time interval. It also may be driven by a Schmitt trigger through timing pulses during an experiment or from a recording.

The program runs in two parts. The first part (SETUP portion) allows the user to define areas of interest on the sampled waveform. The second part (experiment RUN portion) collects samples, looks at the areas defined in the setup, does some preliminary analysis, and displays the data on-the-fly at the terminal. The setup portion collects samples and displays the average waveform for both channels simultaneously on the CRT. The user may then specify time windows (which bracket features of interest on the waveform) by setting cursors on the waveforms displayed.

The user has control over the following parameters:

1. the operating mode, either event-driven or computer-driven
2. the number of signals averaged during the setup portion; the time domain amplification shown on the screen for setup; defining areas of interest on the waveforms, using time windows
3. determining whether voltage measurements will be made relative to zero levels or as absolute differences within the defined time windows (zero levels are continuously updated to negate the effects of DC drift)
4. scaling on the output
5. saving the results by supplying a file name for the data to be stored when asked

All programs are designed in a strictly modular fashion, which allows the quick and easy addition of analysis modules as required by the user. Modules are written in either FORTRAN IV or MACRO (PDP-11 assembler). Certain system routines are used from the RT-11 operating system; thus the use of another operating system requires the replacement of these calls with similar ones from that system.

The programs use named common areas in memory to allow new modules to access only those variables and flags that are needed. This technique effectively "hides" other variables and minimizes side effects.

Precision in the data handling is as follows. An input signal ranging from -5 V to +5 V is passed into a 12-bit A/D converter. This results in a discrimination of 0.024%

provided that the full 10-V range is used. The data stored in the data files reflect this precision. The average waveform plotting is correct to 1% of the maximum range on input due to scaling. The data graphing during the experimental run is correct to 2% of the range specified by the user due to scaling.

HOW TO USE THIS DOCUMENT

General User

A potential user should review the Figures and Output section along with this overview to determine applicability of this package to his/her needs. The user should also consult the hardware and software-module description (see Equipment section below). When the package is assembled and ready to run, the procedure description portion (Section 2) should be kept nearby for reference.

Programmer

All the code necessary to run this task is included in this document in the appendices. The internal documentation of the code and the modularity of the design should facilitate adding to the main code. If more off-line analysis routines are needed, consult Appendix E for the data file structure.

EQUIPMENT

Equipment needed to use the system is listed in the following.

Hardware

Hardware equipment needed is CPU, video terminal with graphics capability, A/D device, Schmitt trigger device, programmable clock, signal source ±5 V, and parallel output port.

Software

Software equipment needed is the RT11 operating system and other program modules (RT11 is a trademark of the Digital Equipment Corp.):

swante.for
swanse.for
swanru.for
colect.mac
getbuf.mac (supplied by AFRRI)
stimul.mac
plot55.mac (supplied by DEC)

CHECKLIST FOR PROPER SETUP OF SYSTEM

1. Ensure all equipment is connected as desired.
2. Turn on the LSI, terminal, disc drive, etc.
3. Boot system.
4. Turn on line clock.
5. Build task if not done yet.
6. Start task (RUN DK1:ALLFOR). The program is up and running if the CRT displays Figure 1.

```
WILL THIS EX. BE SCHMITT TRIGGER DRIVEN? (Y OR N)
```

Figure 1. First question asked when program starts

7. Answer all prompts.

SECTION 2. PROCEDURES FOR OPERATION

SETUP PROCEDURES

Baseline Procedure

The following steps must be performed to calibrate the setup. If the amplifier gain is changed after these steps are performed, the results will not be accurate in magnitude (although still correct in direction).

1. When the line is at 0 mV on the first channel, hit <cr> (carriage return).

2. When the line is at 5 mV on the first channel, hit <cr>.

   NOTE: The <cr> must be hit while the line is at 5 mV; therefore, the longer the standard pulse, the better.

3. Repeat steps 1 and 2 for the second channel.

Amplification Procedure

After drawing the axes, the program will wait for you, the user, to tell it how much time to plot in the screen width. An amplification of 1 will plot 25.6 ms, an amplification of 2 will plot 12.8 ms, an amplification of 3 will plot 8.5 ms, etc. (i.e., 25.6 ms/amplification factor).

You may change the amplification as much as you desire by entering another value each time the amplification prompt comes up. If you hit a <cr> without a value, the program will then proceed to the cursor-placing routine.

Cursor-Placing Procedure

Cursors must be placed on the graphs to delimit time windows. One pair comprises a time window within which the analysis will take place.

Cursors are placed in the following way:

1. Choose a graph as directed.

2. Enter a number (1-256).

3. Hit <cr>. A cursor will now appear at the position requested.

4. Do you want to save this position as a time window delimiter?

   If yes, hit <cr> and then go to step 2.
   If no, go directly to step 2.

5. When you are through placing cursors and defining time windows, enter a 2 and hit <cr>. The cursor-placing procedure is now complete.

RUN PROCEDURES

Scaling Queries

The maximum values must be supplied as asked, including the units. This procedure determines the scaling factors for the output. The number of trials desired must be supplied as asked. The program will now run on its own, either driving the prep with the stimulator or through a Schmitt trigger arrangement. The program will automatically make copies of the data on the screen as necessary.

After all trials have been run, the program will prompt for a name for a file to store the data. Once this is done and the data are stored, the last question the program asks is "Do you wish to run more trials with the same set-up?" If the answer is yes, the program will loop back into the "run" mode again without going through a setup procedure. This feature can be useful in determining those scaling values that give a good display of the data. If the answer is no, the program will end and display all the data files that are stored. At this point, other analysis routines can be used on the stored data. (Please note that Figures 11 and 12 show a sample output.)

SECTION 3. FIGURES AND OUTPUT

This section contains examples of the questions asked to establish parameters and the valid responses to those questions. Also the type of output and displays are shown. The figures included show actual output as seen on the CRT.

SETTING INITIAL PARAMETERS

Figure 2 lists all initial questions that establish the setup parameters. For acceptable answers to the questions, refer to Table 1.

```
WILL THIS EX. BE SCHMITT TRIGGER ORIVEN? (Y OR N)        N
ENTER 2 CHANNELS FOR INPUT :1,2
WHEN CHANNEL 1 IS ZERDED, HIT (CR)
           1         9B7
WHEN 5MV STANDARD APPLIED, HIT (CR)
           1         929
WHEN CHANNEL 2 IS ZERDED, HIT (CR)
           2        1045
WHEN 5 MV STANDARD APPLIED, HIT (CR)
           2         877

# OF SIGNALS TD BE AVERAGED: 3
DD YOU WISH TO DUMP THE BUFFER? (Y DR N) : N
ENTER THE TIME BETWEEN SAMPLES (SEC) : 5
```

Figure 2. Setting initial parameters

Table 1. Acceptable Answers to Questions in Figure 2

Question or Requirement	Acceptable Answer
Number of signals to average	Any number ≥ 3
Enter two channels for input.	Any two channels 0-14 separated by comma. First position refers to stimulating presynaptic channel and upper graph. Second position is postsynaptic channel. Be sure channels selected are ones signals are going into.
Do you wish to dump buffer?	This option allows user to test A/D converter. If chosen, last buffer of digitized data is output to CRT. Program execution then halts.
Enter time between samples.	Enter time as integer. This question will not appear if Schmitt trigger driven.

Each time a sample is taken during the setup portion, the word "WORKING" will appear on the screen (see Figure 3). The number of times it appears corresponds to the number of samples to average. The time between appearances equals the time between samples.

```
#OF SIGNALS TO BE AVERAGEO : 5
OO YOU WISH TO OUMP THE BUFFER?  (Y OR N) :   N
                                       WORKING     1


                                       WORKING     2


                                       WORKING     3


                                       WORKING     4
```

Figure 3. Output seen on CRT while sampling during setup portion

After sampling, the graph will be drawn without plotting (Figure 4). It requires an "amp factor."

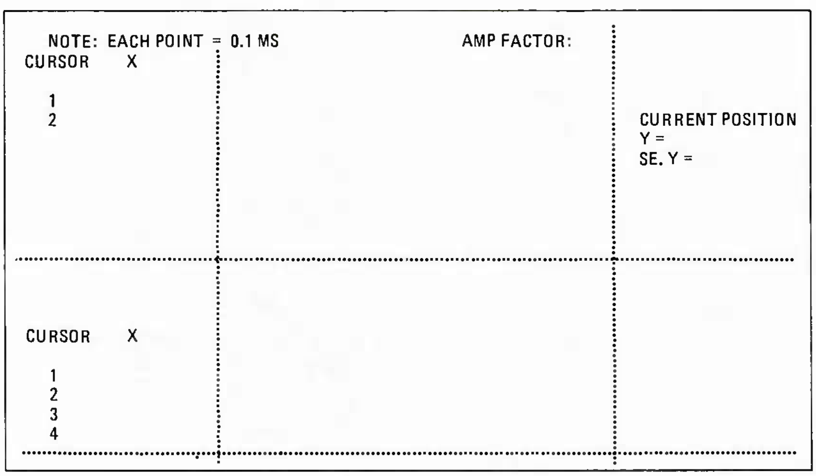

Figure 4. Machine prompts user for an "amp factor."

USING AMP FACTOR OPTION

The "amp factor" option allows the user to expand and amplify the time domain plotted on the screen. Figures 5 through 8 demonstrate the effect of various choices.

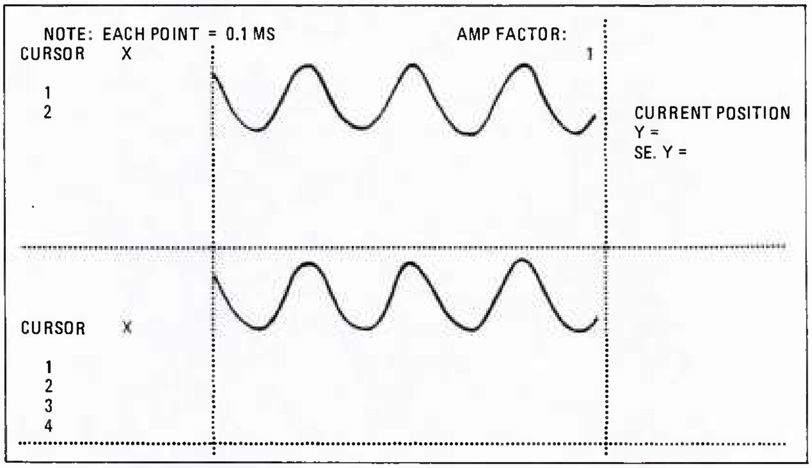

Figure 5. Sinusoidal waveform at amp factor = 1. Compare this plot with Figures 6, 7, and 8 to see how this option works.

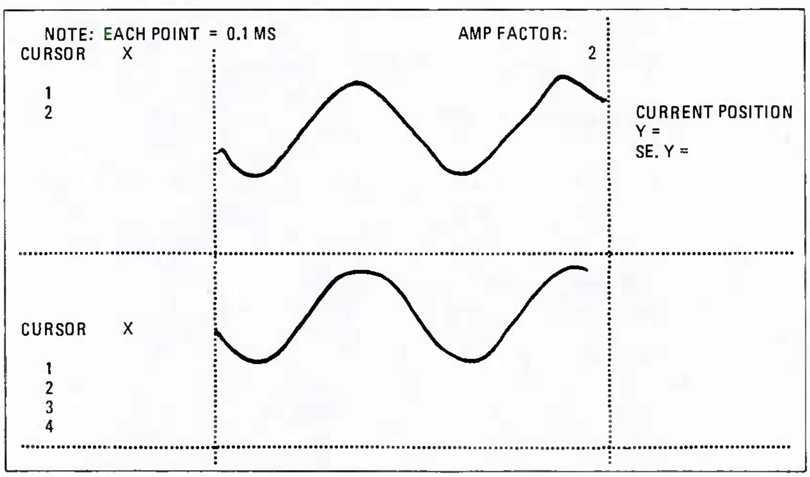

Figure 6. Sinusoidal waveform plot at amp factor = 2. Compare this with Figures 5, 7, and 8.

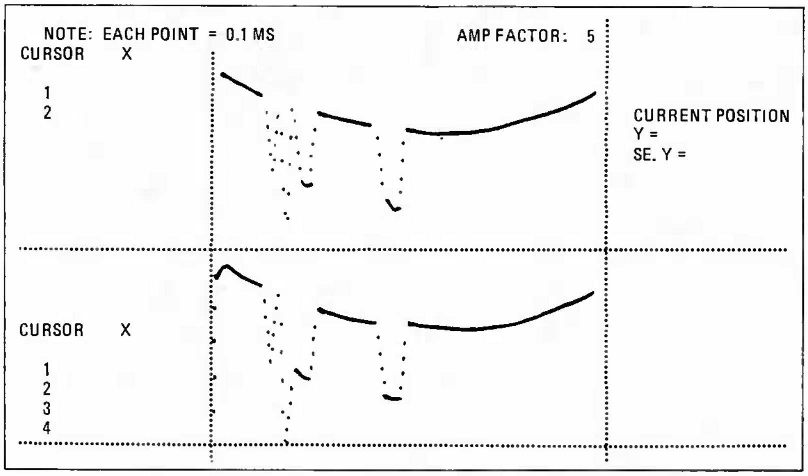

Figure 7. Sinusoidal waveform plot at amp factor = 5. Compare this with Figures 5, 6, and 8.

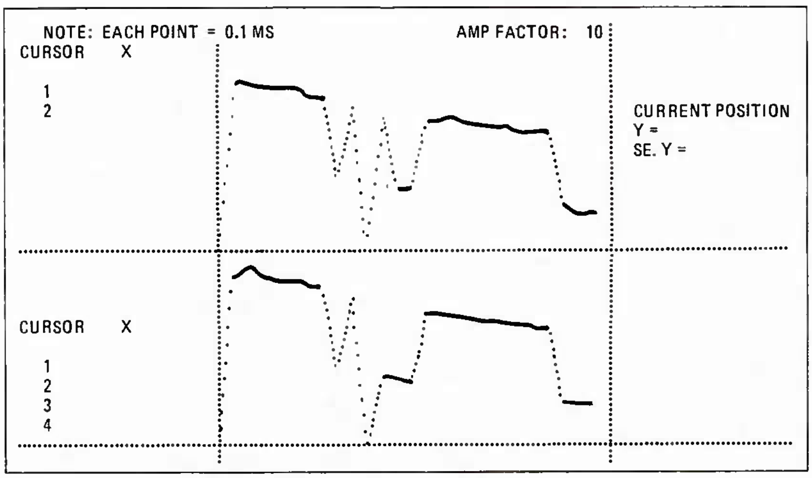

Figure 8. Sinusoidal waveform plot at amp factor = 10. Compare this with Figures 5, 6, and 7.

Now the graph on which to place cursors can be chosen (see Figure 9).

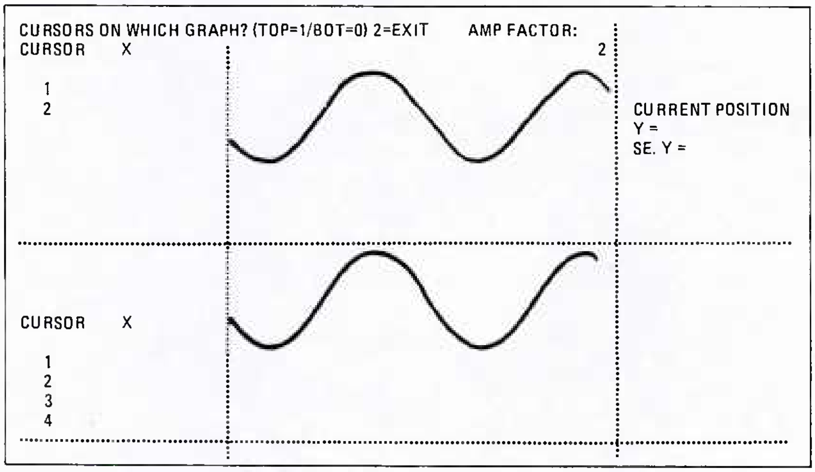

Figure 9. User ready to define time intervals

The cursors have been placed on both graphs, and data are ready to be collected (see Figure 10).

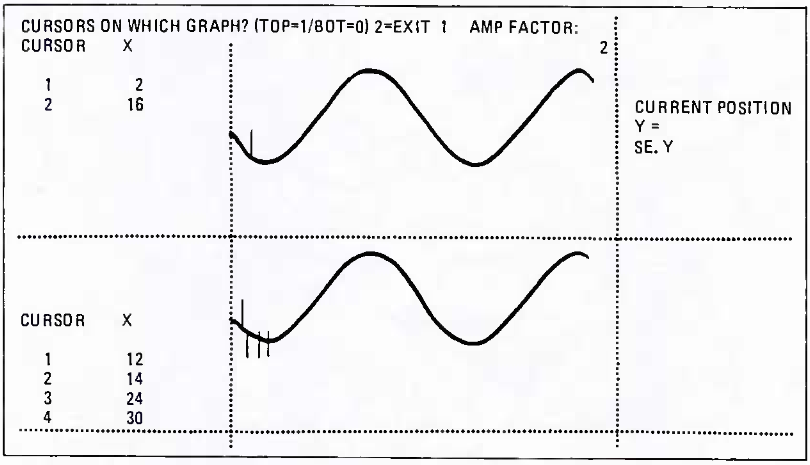

Figure 10. User defining time windows

As the experiment is run, the results are plotted on the CRT as shown in Figures 11 and 12.

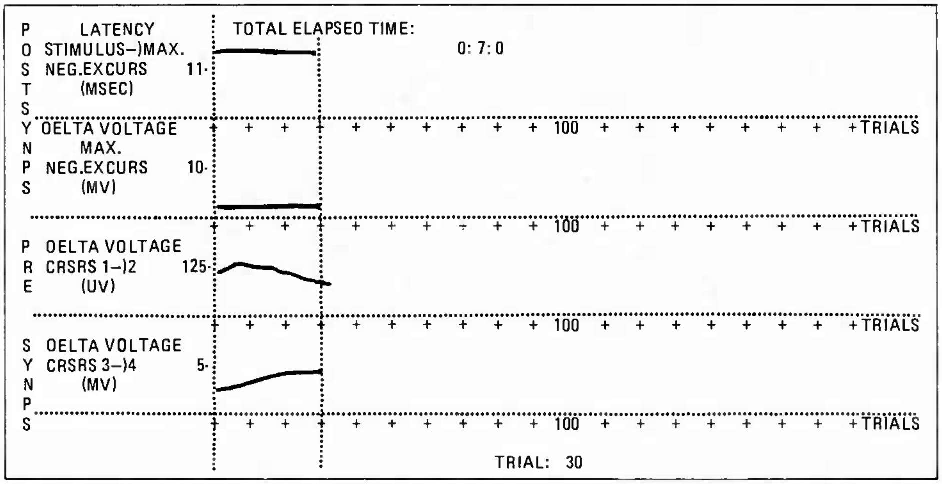

Figure 11. Sample output

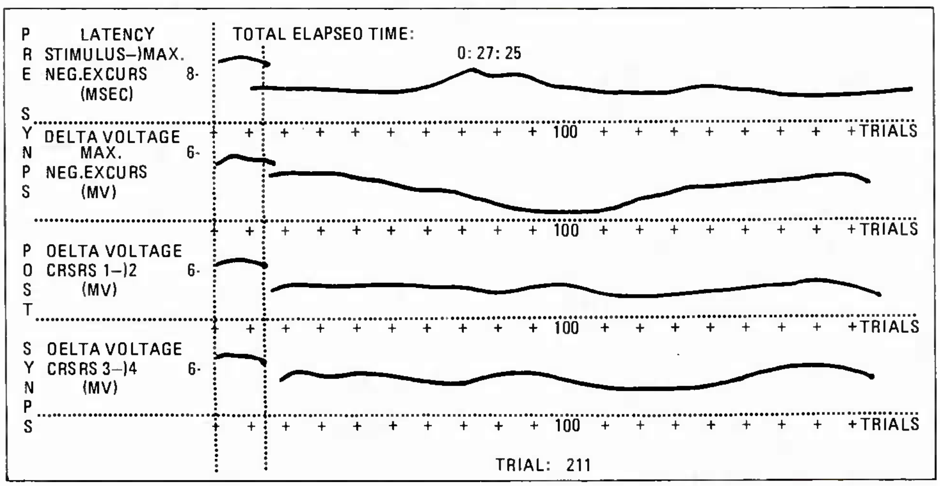

Figure 12. Sample output

APPENDIX A. BUILDING AND LINKING

BUILDING THE TASK

The following sequence is used to compile and link the task:

Compile

FORTRAN

```
FOR/EXTEND/OBJ:DK1:ALLFOR/LIST:DK1:  DK1:(SWANTE+SWANSE+SWANRU)
```

This causes all the FORTRAN programs to be compiled into an object module called ALLFOR. This module is placed on DK1:; it is assumed that the FORTRAN source code is also on DK1:.

macro:

```
MAC/OBJ:DK1:  DK1:GETBUF/LIST:DK1:

MAC/OBJ:DK1:CL  DK1:COLECT/LIST:DK1:

MAC/OBJ:DK1:S  DK1:STIMUL/LIST:DK1:

MAC/OBJ:DK1:PL  DK1:PLOT55
```

The macro programs are compiled separately, and in some cases the names of the object code files are shortened for convenience.

LINKING

```
LINK/EXE:DK1:/MAP:DK1:  DK1:(ALLFOR,CL,PL,GETBUF,S)
```

It is usually convenient to create an indirect command file that performs the tasks outlined above.

APPENDIX B. EXAMPLE OF HARDWARE

The hardware used for the development of this system is:

CPU	-	LSI-11
Terminal	-	VT55-FA
Schmitt trigger/clock	-	KWV-11A
Mass storage	-	RK05
A/D	-	ADV11-A
Parallel output port	-	DRV11

APPENDIX C. LISTINGS FOR FORTRAN PROGRAM

C.1 SWANTE.FOR

```
C
C REAL-TIME INTRA/INTER CELLULAR EVOKED RESPONSE DATA COLLECTION
C PROGRAM.  MAY BE SCHMITT TRIGGER DRIVEN OR IT MAY DRIVE THE
C PREP ACCORDING TO A TIME PARAMETER WHICH IS PROMPTED FOR.
C
C
C        MASTER PROGRAM
10      CALL SETUP
        CALL PLOT55(9,0,0)
        CALL PLOT55(10,,)
        CALL PLOT55(9,35,11)
        CALL PLOT55(12,,'END OF SET UP;HIT <CR> TO CONTINUE')
        CALL PLOT55(9,35,14)
        CALL PLOT55(12,,'ENTER S TO RESTART')
        CALL PLOT55(2,512,)
        ACCEPT 1000, S
        IF (S.EQ.'S') GO TO 10
        CALL RUNIT
1000    FORMAT(A1)
        END
C ------------------------------------------------------------
C
```

C.2 SWANSE.FOR - SET-UP PORTION

```
c
c       25 march 80
c
c       routine: setup
c
c       routine for the initial set-up of the experiments
c


c       this routine receives an initial 5mv square pulse
c       and calculates the function that maps the real
c       voltage across the membranes from the digitized
c       input. it then uses this value later to output
c       the potential.
c
c       the parameters passed include the slope 'm' and
c       the y intercept 'b' of this function.
c
c       following this action, the program receives and
c       averages 'n' signals and presents these vs time.
c       the experimenter then places four cursors along
c       the signal trace to indicate the points between
c       which he wishes the dv measured.
c
c                                 main prog
c                                /
c                     _ setup _          runit
c                   /     |
c          stdsig      sigavg      csrplc ___
c         /        /    /
c   colect   getbuf   axes    trcplt      store     disp
c                                /
c                          scale       plot
```

```
c
        subroutine setup
c
        call plot55(2,1+2+4+32+64+128+256+512,) !enable graphics
        call plot55(13,72,)      ! clear
        call plot55(13,74,)      !         screen
        call stdsig(1)  ! find mapping function
        call stdsig(2)
        call plot55(13,72,)
        call plot55(13,74,)
        call sigavg      !iampl is defined in trcplt
        call csrplc
        call copy        !copy screen for later reference
c
c       set up is now complete
        return
        end
c
c       ___________________________
c       ___________________________
c
c       routine for using a standard signal to generate
c       the mapping function for the digitized values to
c       their mv equiv.(for output later)
c
        subroutine stdsig(i)
        common /scalar/ic(2,4),ib(2),m(2)
        common /schmit/ist,izzz
        common ichtb
        integer volt0,volt2
        byte ichtab(2)
        equivalence(ichtab,ichtb)
        if (i.eq.2) go to 20     !skip if already answered
10      write (5,1000)
1000    format('$will this ex. be schmitt trigger driven? <y or n>  :',$)
        accept 1500,ist
1500    format (1a1)
        if(ist.ne.'y'.and.ist.ne.'n')go to 10
        if (ist.eq.'y') ist=1
        if (ist.eq.'n') ist=0
        write(5,2000)
2000    format('$enter 2 channels for input :')
        accept 2500, ichtab(1),ichtab(2)
2500    format(2i3)
20      write (5,3000)i
3000    format ('$when channel',i3,' is zeroed, hit <cr>')
        accept 3500
3500    format (a1)
        j=ichtab(i)
        call colect(volt0,j)     !get sample from a/d @ 0 mv
        write(5,4000)j,volt0
4000    format(2i10)
        write(5,4500)
4500    format ('$when 5mv standard applied, hit <cr>')
        accept 3500
        call colect(volt2,j)     !get sample from a/d @ 5 mv
        write(5,4000)j,volt2
        do 30 ii=1,32000
30      continue                 !pause for a moment
        itry=.2 * (volt2-volt0)
        if (itry.eq.0)itry=1
        m(i)=itry
        ib(i)=volt0
        return
        end
c       ________________________
c       ________________________
c
```

FORTRAN Program

```
          subroutine sigavg
c
c         routine to receive, average and display a signal
c         from the a-d to provide a trace to place the
c         cursors.
c
          integer*4 time1,time2
          integer hrs,mins,secs,tcks
          byte ichtab(2)
          real array1(256), array2(256)
          real sx1(256),sx2(256),sx21(256),sx22(256)
          common /scalar/ic(2,4),ib(2),m(2)
          common /datbuf/ibufr(512),ibufr1(256),ibufr2(256)
          common /stderr/se1(256),se2(256)
          common /schmit/ist,izzz
          common ichtb
          equivalence(ichtb,ichtab)
c
c         variables: sx1= sum of x1
c                    sx21= sum of x1**2
c                    sx2= sum of x2
c                    sx22= sum of x2**2
          call plot55(9,0,0)
          call plot55(13,74,)
          write(5,1000)
1000      format('$  of signals to be averaged:')
          accept 1500, iternm
1500      format(i10)
c
          write(5,2000)
2000      format('$do you wish to dump the buffer? <y or n>:')
          accept 2500,idump
2500      format(a1)
c
          if (ist.eq.1) go to 10  !skip if schmidt triggered
          write(5,3000)
3000      format('$enter the time between samples (sec):')
          accept 3500, izzz
3500      format(i5)
c
10        do 20 i=1,256
          array1(i)=0
          array2(i)=0
          sx1(i)=0
          sx21(i)=0
          sx2(i)=0
          sx22(i)=0
20        continue
c
c         collect 'iternm' samples and avg them
c
          icount=256
          ichcnt=2
          krt=3
          kcnt=1
          ich1=ichtab(1)
          ich2=ichtab(2)
          do 90 itr=1,iternm
          call gtim(time1)
          call colect(ib(1),ich1)         !find current zero level
          call colect(ib(2),ich2)         !find current zero level
          if (ist.eq.0) call stimul  !if not schmitt trig. run trial
          call getbuf(ibufr,icount,ichtab,ichcnt,krt,kcnt,ist,ierr)
          write(5,4000)itr
4000      format(35x,'working ',i4////)
c         now write these values into two arrays which represent
c         the two channels of input.
```

```
c
c
        i=1
        j=1
        ibufr(1)=ibufr(3)        !first point is garbage
        ibufr(2)=ibufr(4)
30      if (j.gt.256) go to 40
        reali=ibufr(i)
        array1(j)=array1(j)+(ibufr(i)-ib(1))
        sx1(j)=sx1(j)+(reali-ib(1))/m(1)
        sx21(j)=sx21(j)+((reali-ib(1))/m(1))**2
c
        reali=ibufr(i+1)
        array2(j)=array2(j)+(ibufr(i+1)-ib(2))
        sx2(j)=sx2(j)+(reali-ib(2))/m(2)
        sx22(j)=sx22(j)+((reali-ib(2))/m(2))**2
c
        j=j+1
        i=i+2
        go to 30
c
40      if (ist.eq.1) go to 90  !skip if s.trig
        call cvttim(time1,hrs,mins,secs,tcks)
50      call gtim(time2)
        call cvttim(time2,ihrs,imins,isecs,itcks)
        if(isecs.eq.secs.and.itcks.eq.tcks)write(5,4500)
4500    format(' turn on the line clock')
        if((secs-isecs).lt.izzz.and.(isecs-secs).lt.izzz) go to 50
c
c       go get next sample
90      continue
        ap=iternm
        do 60 i=1,256
        se1(i)=sqrt(abs(((sx21(i)/ap)-((sx1(i)/ap)**2)/ap)))
        se2(i)=sqrt(abs(((sx22(i)/ap)-((sx2(i)/ap)**2)/ap))) !s.e.
60      continue
c
        if(idump.ne.'y') go to 80
                do 70 i=1,512,2
70              write(5,5000)i,ibufr(i),ibufr(i+1)
5000            format(i5,i10,i10)
                write(5,5500)m(1),ib(1),m(2),ib(2)
5500    format(' m(1)=',i5,3x,'ib(1)=',i5,//' m(2)=',i5,3x,'ib(2)=',i5)
                stop
c       plot the average
80      call axes
        call trcplt(array1,array2,iternm)
        return
        end
c       ----------------------
c       ----------------------
c
        subroutine trcplt(ar1,ar2,itq)
c
c
c
        dimension ar1(256),ar2(256)
        common /datbuf/ibufr(512),ibufr1(256),ibufr2(256)
        common /amplfy/iamp2
        common ichtb,ilat
```

FORTRAN Program

```
c
        ah=itq
        do 10 k=1,256
        ibufr1(k)=ar1(k)/ah
        ibufr2(k)=ar2(k)/ah
10      continue
20      call plot55(9,45,0)
        call plot55(11,,)
        call plot55(12,,' amp factor : ')
        call plot55(13,72,)
        do 15 iaia=1,59
15      call plot55(13,67,)
        accept 1000, iampl
1000    format(i3)
        if (iampl.eq.0)return
        iamp2=iampl
c
        i2=10   !signal to plot on lower graph
        call scale(ibufr1,256,r1,r2,ilat)
        call plot55(1,0,)
        call plot55(7,111,10)
        call plot(ibufr1,256,r1,r2,i2,iampl)
c
        i2=115  !signal to plot on upper graph
        call scale(ibufr2,256,r1,r2,ilat)
        call plot55(1,1,)
        call plot55(7,111,115)
        call plot(ibufr2,256,r1,r2,i2,iampl)
        go to 20
        end
c       ----------------------
c
c       routine to plot the data
c
        subroutine plot(i12,i1,r1,r2,i2,iampl)
c
c       variable names consistent with 'scale'
c
        integer i12(256)
        real r(256)
        if (r1.ne.r2) go to 5
        write(5,1000)
1000    format(' faulty signal read',/25x,'terminate execution')
        stop
5       r1=100/(r1-r2)  !scale data to fit plot size
        j=0
        do 10 i=111,366,iampl
        j=j+1
        r(j)=i12(j)
10      call plot55(8,i,ifix((r(j)-r2)*r1)+i2)
        return
        end
c
c       -----------------------
c       ----------------------
```

```
c
c       this routine will scale the data arrays for
c       plotting. it returns r1=max value; r2=min
c       value. r is the input array.
c       i1 is the   of points.
c
        subroutine scale(i12,i1,r1,r2,ilat)
        integer i12(256)
        dimension r(256)
        do 10 i=1,256
10      r(i)=i12(i)
        r1=-1.0e32
        r2= 1.0e32
        do 30 i=1,i1
        if (r(i).gt.r1) r1=r(i)
        if (r(i).lt.r2) go to 20
        go to 30
20              r2=r(i)
                ilat=i  ! use as flag for location of stimulus artifact
30      continue
        return
        end
c
c       ________________________
c       _____________________________
c
c       routine to place cursors on the traces shown.  four
c       cursors can be placed on the bottom graph, two
c       cursors are placed on the top trace.
c
        subroutine csrplc
        common /datbuf/ibufr(512),ibufr1(256),ibufr2(256)
        common /stderr/se1(256),se2(256)
c
c       decide which graph to set cursors on.
10      call plot55(9,0,0)
        call plot55(11,,)
        call plot55(12,,'cursors on which graph?(top=1/bot=0) 2=exit: ')
        accept 1000, igraph
1000    format(i1)
        if (igraph.eq.2) go to 70
        call plot55(9,0,0)
        call plot55(11,,)
        call plot55(1,igraph,)
c
        if (igraph.eq.0) nc=4    !if graph0, get 4 cursors
        if (igraph.eq.1) nc=2    !if graph1, get 2 cursors
c
c       identify and store the cursors in 'ic'
        do 60 i=1,nc
c       prompt for cursor position
20      call plot55(9,0,0)
        call plot55(11,,)
c       256=25.6 ms ie: crs position= 10* time
        call plot55(12,,'cursor position(1-256) use <cr> to store: ')
        call plot55(9,43,0)
        accept 1500, icp
1500    format(i3)
        if (icp.gt.400) go to 40
        if (icp.lt.0) go to 40
        if (icp.eq.0) call store(i,igraph,ilast)
        go to 50
40      call plot55(9,0,0)
        call plot55(11,,)
        call plot55(12,,'error on input, hit <cr> to continue')
        accept 2000, iii
2000    format(a1)
        go to 20
50      if (icp.eq.0) go to 60
        call plot55(6,ilast+110,0)
        call plot55(6,icp+110,1)
        if (igraph.eq.0)call disp(ibufr1,se1,icp,igraph+1)
        if (igraph.eq.1)call disp(ibufr2,se2,icp,igraph+1)
```

```
        ilast=icp
        go to 20
60      continue
        go to 10
70      return
        end
c
c       ______________________________
c       this routine enters the cursor values into the
c       table 'ic' which holds the values for later use.
c       as well, the values stored are displayed on the
c       screen as well as the cursor positions.
c
        subroutine store(i,igraph,ilast)
        common /scalar/ic(2,4),ib(2),m(2)
        common /amplfy/iamp2
c
        ic((igraph+1),i)=ilast/iamp2    !the last value, before a
c                                        zero was entered to exececute
c                                        a save, is placed in ic(i,j)
c                                        where 'i' is the graph  and
c                                        'j' is the jth cursor stored.
c
c       now , display the saved value in the app.spot
c
        iplace=1-igraph
        call plot55(9,9,((15*iplace)+2)+i)
        write(5,1000)ilast
1000    format(i4)
        ilast=0
        return
        end
c       _________________________
c       _________________________
c
        subroutine axes
c
c
        call plot55(9,0,0)
        call plot55(10,,)
        call plot55(12,,'   note: each point = 0.1 ms')
        call plot55(4,1,115)
        call plot55(4,1,10)
        call plot55(5,370,1)
        call plot55(5,110,1)
        call plot55(9,1,1)
        call plot55(12,,'cursor     x')
        call plot55(9,3,3)
        call plot55(12,,'1')
        call plot55(9,3,4)
        call plot55(12,,'2')
c
        call plot55(9,1,16)
        call plot55(12,,'cursor     x')
        call plot55(9,3,18)
        call plot55(12,,'1')
        call plot55(9,3,19)
        call plot55(12,,'2')
        call plot55(9,3,20)
        call plot55(12,,'3')
        call plot55(9,3,21)
        call plot55(12,,'4')
        call plot55(9,64,4)
        call plot55(12,,'current position')
        call plot55(9,64,5)
        call plot55(12,,'y=    ')
        call plot55(9,64,6)
        call plot55(12,,'se.y=')
        return
        end
c
c       _____________________________
c       _____________________________
```

```
c
c
        subroutine disp(r,rx,iqr,iqx)
        dimension r(256),rx(256)
        common /scalar/ic(2,4),ib(2),m(2)
        call plot55(9,68,5)
        write(5,1000)r(iqr)/m(iqx)
        call plot55(9,68,6)
        write(5,1000)rx(iqr)
1000    format(f10.3)
        return
        end
c
c       ____________________________
```

C.3 SWANRU.FOR - EXPERIMENT RUN SECTION

```
c       subroutine call mapping:
c                       main program
c                      /
c               swanset           runit
c                                /
c                    graph _   runex  _
c                         /      |
c                    getbuf    tvmax       negmax
c

        subroutine runit
        byte ifile(14)
        byte blank
        common /scalar/ic(2,4),ib(2),m(2)
        common /units/iunits,iunit2
        common /outfil/laten(1000),mvolt(1000),negex1(1000),negex2(1000)
        common ichtb,ilat,idif,itest
        data blank/' '/
10      do 20 i=1,14
20      ifile(i)=blank
        call plot55(2,1+2+4+32+64+512,)
c
        call plot55(9,0,0)
        call plot55(10,,)
        write(5,950)
950     format('$differences will be relative to zero(0) or 1st cursor(1): ')
        accept 960, idif,itest
960     format(2i2)
        write(5,1000)
1000    format('   supply the information requested on the max. values')
1500    format(///,20x,'postsynaptic',//,'$maxneg cursors 1-2 ( ,[mv/uv]) : ')
25      write(5,1500)
        accept 2000,is1,iunits
        if(iunits.ne.'uv'.and.iunits.ne.'mv')go to 25
        if(iunits.eq.'uv')iunits=1000   !scale change to microvlt
        if(iunits.eq.'mv')iunits=1
2000    format(i6,a2)
27      write(5,2500)
2500    format(//,20x,'presynaptic',//,' maxneg cursors 1-2 ( .[mv/uv] : '.$)
        accept 2000,is2,iunit2
        if(iunit2.ne.'uv'.and.iunit2.ne.'mv')go to 27
        if(iunit2.eq.'uv')iunit2=1000
        if(iunit2.eq.'mv')iunit2=1
3000    format(i5)
        write(5,3500)
3500    format(//,' maxneg ex(mv);cursors 3-4: ',$)
        accept 2000,is3
        write(5,4000)
4000    format(/,'$how many trials? (1000 max):')
        accept 3000,iiiii
        call graph(is1,is2,is3)
        call runex(is1,is2,is3,iiiii)
```

```
c
c       write the results for today into a storage file
        call plot55(9,0,0)
        write(5,4500)
4500    format(5x,$'enter file name for storage- dk1:<date>.dat :')
        accept 5000,ifile
5000    format(14a1)
        open (unit=1,name=ifile,type='new')
        do 30 i=1,iiiii
        aa=float(mvolt(i))/float(m(2))
        ab=float(negex1(i))/float(m(1))
        ac=float(negex2(i))/float(m(1))
        write(1,5500)laten(i),aa,ab,ac
30      continue
5500    format(i5,3f10.4)
        iflag=-5
        write(1,5500)iflag,aa,ab,ac
        close(unit=1,dispose='save')
        call plot55(9,0,0)
        call plot55(10,,)
        write(5,6000)
6000    format('$do you wish to run more trials with the same setup?')
        accept 6500,iask
6500    format(a1)
        if(iask.eq.'y') go to 10
        return
        end
c       -----------------------------------------------------
c       -----------------------------------------------------
c       this routine sets up the graphics display for
c       the experiment
c
        subroutine graph(is1,is2,is3)
        common /scalar/ic(2,4),ib(2),m(2)
        common /units/iunits,iunit2
        common ichtb,ilat
        call plot55(9,0,0)
        call plot55(10,,)
        call plot55(9,20,0)
        call plot55(12,,'total elapsed time:')
        call plot55(4,1,30)
        call plot55(4,1,80)
        call plot55(4,1,129)
        call plot55(4,1,130)
        call plot55(4,1,180)
        call plot55(5,111,1)
        do 20 j=5,20,5
        do 10 i=18,79,3
        call plot55(9,i,j)
10      call plot55(12,,'+')
20      continue
        do 30 i=5,20,5
        call plot55(9,74,i)
        call plot55(12,,'trials')
        call plot55(9,47,i)
30      call plot55(12,,'100')
        call plot55(9,12,2)
        write(5,1000)(ic(2,2)-ilat)/9   !1/2 of the scaled values
        call plot55(9,12,6)
        write(5,1000)is1/2
1000    format(i6,'')
        call plot55(9,12,11)
        write(5,1000)is2/2
        call plot55(9;12,16)
        write(5,1000)is3/2
        call plot55(9,0,0)
        call plot55(12,,'p      latency')
        call plot55(9,0,1)
        call plot55(12,,'o stimulus->max.')
        call plot55(9,0,2)
        call plot55(12,,'s neg.excurs')
```

```
      call plot55(9,0,3)
      call plot55(12,,'t  (msec)')
      call plot55(9,0,4)
      call plot55(12,,'s')
      call plot55(9,0,5)
      call plot55(12,,'y delta voltage')
      call plot55(9,0,6)
      call plot55(12,,.'n    max.')
      call plot55(9,0,7)
      call plot55(12,,'p neg.excurs')
      call plot55(9,0,8)
      if(iunits.eq.1)call plot55(12,,'s   (mv)')
      if(iunits.ne.1)call plot55(12,,'s   (uv)')
      call plot55(9,0,11)
      call plot55(12,,'p delta voltage')
      call plot55(9,0,12)
      call plot55(12,,'r crsrs 1->2')
      call plot55(9,0,13)
      if(iunit2.eq.1)call plot55(12,,'e     (mv)')
      if(iunit2.ne.1)call plot55(12,,'e     (uv)')
      call plot55(9,0,16)
      call plot55(12,,'s delta voltage')
      call plot55(9,0,17)
      call plot55(12,,'y crsrs 3->4')
      call plot55(9,0,18)
      call plot55(12,,'n     (mv)')
      call plot55(9,0,19)
      call plot55(12,,'p')
      call plot55(9,0,20)
      call plot55(12,,'s')
      call plot55(9,42,22)
      call plot55(12,,'trial:')
      return
      end
c
c     ---------------------------------------------------------
c     ----------------------------------------------------------------
c
c     routine to run the experiments and to collect
c     and display the data on the graphs
c
      subroutine runex(is1,is2,is3,iiiii)
c
      byte ichtab(2)
      common /datbuf/ibufr(512),ibufr1(256),ibufr2(256)
      common /scalar/ic(2,4),ib(2),m(2)
      common /howmny/iter,itern
      common /schmit/ist,izzz
      common ichtb,ilat
      integer*4 time1,time2
      equivalence(ichtb,ichtab)
c
c     variables:
c            ic      the table of cursor values
c            ib,m    the parameters for output transform
c            iter    trial counter
c            itern   trial counter for display control
c     a      laten   array for storing latency values ,ch 2
c            mvolt   array for storing max. neg. values,ch 2
c            negex1  array for storing max. neg. values,1st curs.,ch 1
c            negex2  array for storing max. neg. values,2nd curs.,ch 1
c            ibufr1  data buffer, ch 1
c            ibufr2  data buffer, ch 2
c            time1,time2     current time for sampling control
c            izzz    time between samples (snoozzzzz time)
```

FORTRAN Program

```
c
                 iter=0  !set initial value
                 itern=1
                 icount=256
                 ichcnt=2
                 krate=3
                 kcount=1
                 ich1=ichtab(1)
                 ich2=ichtab(2)
c
10        call gtim(time1)
          call cvttim(time1,ihrs,imin,isec1,itck)
c
c         run clock at 10khz, convert at ea. overflow
          if(ist.eq.0) call stimul !run the trial if not schmitt trig.
          call colect(ib(1),ich1)         !get current zero level
          call colect(ib(2),ich2)         !get current zero level
          call getbuf(ibufr,icount,ichtab,ichcnt,krate,kcount,ist,ierr)
          i=1
          j=1
          ibufr(1)=ibufr(3)
          ibufr(2)=ibufr(4)        ! first points are junk
18        if(i.gt.256)go to 20
          ibufr1(i)=ibufr(j)
          ibufr2(i)=ibufr(j+1)     !separate both channels
          j=j+2
          i=i+1
          go to 18
20        iter=iter+1
          call plot55(9,50,21)
          write(5,1000)iter
1000      format(i5)
c
          if (iter.eq.iiiii ) go to 50
          if (iter.eq.199) call copy
          if (iter.eq.399) call copy
          if (iter.eq.599) call copy
          if (iter.eq.799) call copy
          call plot55(5,itern+111,0)
          call plot55(5,itern+113,1)
c
          call tvmax(ibufr2,is1)  !process ch 2
          call negmax(ibufr1,is2,is3)     !process ch 1
          call plot55(9,0,23)
          call plot55(10,,)
          itern=itern+2
c
c         wait for izzz seconds to elapse before running the next trial
30        call gtim(time2)
          call cvttim(time2,ihrs,imin,isec2,itck)
          if(ist.eq.1) go to 40 !skip the wait if schmitt triggered
          if ((isec2-isec1).lt.izzz.and.(isec1-isec2).lt.izzz) go to 30
40        call plot55(9,41,0)
          writae(5,1500)ihrs,imin,isec2
1500      format(i2,':',i2,':',i2)
          go to 10
c
50        call copy
          return
          end
c-----------------------------------------------------------------------
c
          subroutine copy
          common /howmny/iter,itern
          call plot55(9,79,23)
          call plot55(13,93,)
          itern=1
          return
          end
c
c_____________________________________
c         --------------------------------------------------------------
```

```
c
c       this routine uses the cursors designated for the post-
c       synaptic side to find the most negative value sampled
c       between the cursors and the time when this value occured.
c       this time is measured from the initial depol. and is
c       displayed as a latency.
c
        subroutine tvmax(istore,is1)
        dimension istore(256)
        common /scalar/ic(2,4),ib(2),m(2)
        common /outfil/laten(1000),mvolt(1000),negex1(1000),negex2(1000)
        common /howmny/iter,itern
        common /units/iunits,iunit2
        common ichtb,ilat,idif,itest
        ivmin=10000
c
        do 10 i=ic(2,1),ic(2,2) !look between cursors
        if(itest.eq.1)write(5,1000)istore(i)
1000    format(i10)
        if (istore(i).gt.ivmin) go to 10
                ivmin=istore(i) !difference relative to zero
                lat=i
10      continue
        if(itest.eq.1)write(5,1500)ivmin,lat
1500    format(' minimum=',i10,' at lat =',i10)
        if(idif.eq.1)ivmin=istore(ic(2,1))-istore(lat) !diff rel to 1st curs
        if(ivmin.lt.0)ivmin=-ivmin
c
        laten(iter)=float(lat-ilat)/4.65
        if((lat.ne.ic(2,1)).and.(lat.ne.ic(2,2)))go to 20
                lat=ilat
                ivmin=0
20      mvolt(iter)=ivmin
c               graph it now
        iy1=((50./(float(ic(2,2)-ilat)/4.65))*(float(lat-ilat)/4.65))+180.
        if (iy1.gt.230)iy1=230
        if (iy1.lt.180)iy1=180          ! check scaling
c
        iy2=(50./float(is1))*iunits*(float(ivmin)/float(m(2)))+130.
        if (iy2.gt.180)iy2=180
        if (iy2.lt.130)iy2=130
        call plot55(1,1,)
        call plot55(3,110+itern,iy1)
        call plot55(3,111+itern,iy2)
        return
        end
c
c       ------------------------------------------------------------
c       ----------------------------------------------------------
c
c       this routine finds the most neg. points within the time
c       windows provided by cursor sets 1-2  3-4. the input
c       examined is from channel 1, presynaptic.
c
        subroutine negmax(istore,is2,is3)
        dimension istore(256)
        common /scalar/ic(2,4),ib(2),m(2)
        common /outfil/laten(1000),mvolt(1000),negex1(1000),negex2(1000)
        common /howmny/iter,itern
        common /units/iunits,iunit2
        common ichtb,ilat,idif,itest
c
        do 20 j=1,3,2
        ivmin=10000
        do 10 i=ic(1,j),ic(1,j+1)
        if(itest.eq.1)write(5,1000)istore(i)
1000    format(i10)
        if (istore(i).ge.ivmin) go to 10
                        ivmin=istore(i)
                        lat=i
```

```
10      continue
        if(itest.eq.1)write(5,1500)ivmin,lat
1500    format(' minimum =',i10,' lat=',i10)
        if(idif.eq.1)ivmin=istore(ic(1,j))-istore(lat)
        if(ivmin.lt.0)ivmin=-ivmin
c
        if (j.lt.2) negex1(iter)=ivmin
        if (j.gt.2) negex2(iter)=ivmin
        call plot55(1,0,)
        if (j.gt.2) go to 30
c
        if(lat.eq.ic(1,j+1).or.lat.eq.ic(1,j)) ivmin=0
c                                      !if the min pt.@ curs., defined as 0
        iz1=(50./float(is2)*iunit2)*((float(ivmin)/float(m(1))))+80.
        if(iz1.gt.130)iz1=130
        if(iz1.lt.80)iz1=80
        call plot55(3,110+itern,iz1)
        go to 20
c
30      iz2=(50./float(is3))*((float(ivmin)/float(m(1))))+30.
        if(iz2.gt.80)iz2=80
        if(iz2.lt.30)iz2=30
        call plot55(3,111+itern,iz2)
c
20      continue
        return
        end
```

APPENDIX D. LISTINGS FOR MACRO PROGRAM

D.1 COLECT.MAC

```
                              .title colect
                              .ident /don01/
                              ;
                              ;
                              ; d.o. norman
                              ;
                              ; routine to take a single a/d on a
                              ; given channel
                              ;
                              ; fortran calling procedure
                              ; call colect(dataword,channel)
                              ; dataword ::: receives the result of the a/d
                              ; channel  ::: input channel
                              ;
        adsr = 170400
        adbr = 170402
colect::
start:  clr     @ adsr          ;clear a/d stat reg
        mov     2(r5),r0        ;set up address for storage
        mov     @4(r5),r1       ;get channel number
        asl     r1
        asl     r1              ; shift it
        asl     r1              ;     into the
        asl     r1              ;        correct
        asl     r1              ;                position
        asl     r1              ; before
        asl     r1              ;        'oring' it
        asl     r1              ;                with the adsr
        bis     r1,@ adsr       ; set up channel
;
        inc     @ adsr          ; start conversion
wait:   tstb    @ adsr          ; done?
        bpl     wait            ; if not then wait
        mov     @ adbr,(r0)     ; load data word
        rts     pc
        .end
;-------------------------------------------------------------
---
```

D.2 STIMUL.MAC

```
;-------------------------------------------------------------
---
;
                              .title stimul
                              .ident /don02/
                              ; d.o. norman
                              ;
                              ; routine to cause a triggering of
                              ; the prep stimulator through the
                              ; external triggering connection.
                              ;
                              ; assumes a drv-11 (parallel output)
                              ; is used and set up according to
                              ; specs.
                              ;
                              ; trigger is through bit 0
                              ;
                              ; fortran call:
                              ; call stimul
                              ;
```

```
        drvout= 167772
stimul::
        clr     @ drvout
        inc     @ drvout        ; toggle bit 0 (turn it on)
        nop
        nop
        nop                     ; wait a while
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        clr     @ drvout        ; clear bit 0 (turn it off)
        rts     pc
        .end
;-----------------------------------------------------------
-
;-----------------------------------------------------------
-
        .title  getbuf - rt11 adv/kwv driver module - program control
;;;     .sbttl  getbf1 - burst mode sampling version
        .ident  /lmf01a/
;
        .nlist  cnd
        .enabl  lc
;
;       .mcall
;
;written:
;       l. m. fraser    21-mar-80
;
; call syntax:    ...fortran
;
;       call
 getbuf(ibuf,icount,ichtab,ichcnt,krate,kcount,iwait,ierr)
;
; where:
;       ibuf    = integer buffer to receive data in order as sampled.
;                       must be dimensioned icount*ichcnt long
;       icount  = number of sample sets to convert
;       icntab  = *byte* table of channel numbers in order of
 interest.
;       ichcnt  = length of channel list
;       krate   = rtc clock count units parameter (1 = mhz) see users
 manual
;       kcount  = number of units above between overflows
;       iwait   = wait for st2 trigger if >< 0
;       ierr    = error indicator (reflects only data overruns)
;
;
; define parameter below to select level of error indication
;
;       errmrk not defined:     no error checking
;       errmrk = 0              count overflows only
;       errmrk = 1              count errors  set data point bit 12 on error
;
;       errmrk = 1
;
; local macros
;
; parameter error reporter - default error 81. "invalid
 argument"
;

        .macro prmerr prm
        mov     prm,prm.8(r5)
        mov     spsave,sp
        return                  ; return with error report
        .endm   prmerr
```

```
; get address from fortran argument list or check of loc
 blank (ok if ne)
;
        .macro geta off,loc,?a,?b
        .iif idn loc,r5 .error  ;you're screwing up the parameter link
 (r5)
        cmpb     off/2,(r5)
        ble     a
        prmerr   off
a:
        .if nb loc
        mov     off(r5),loc
        cmp     loc, -1
        bne     b
        trap    128.+16.
        .iff
        cmp     off(r5), -1
        .endc
b:
        .endm   geta
;
; get parameter from fortran argument list - word arguments
 only
;
        .macro  getp off,loc
        geta    off,loc
        mov     @loc,loc
        .endm   getp
;
; put parameter to fortran parameter list
;
        .macro  putp off,loc,?a,?b
        cmpb     off/2,(r5)
        ble     b
        prmerr   off
b:      cmp     off(r5), -1
        bne     a
        prmerr   off
a:
        .if nb loc
        mov     loc,off(r5)
        .iff
        clr     off(r5)
        .endc
        .endm
.page
;
; local definitions.......
;
; define fortran parameter offsets
;
prm.1    = 2
prm.2    = 4
prm.3    = 6
prm.4    = 10
prm.5    = 12
prm.6    = 14
prm.7    = 16
prm.8    = 20
;
; define adv11 and kwv11 address
;
adcsr    = 170400
asbuf    = adcsr + 2
kwcsr    = 170420
kwbpr    = kwcsr + 2
;
```

```
; define priority mask
;
pr0     = 0
pr7     = 340
;
; define adv csr bit offsets
;
ad.go   = 1      ; start immediately
ad.mad  = 4      ; maint - load all data from mux bit 0
ad.ide  = 10     ; set identify mode
ad.exe  = 20     ; external start enable
ad.cle  = 40     ; clock overflow start enable
ad.ine  = 100    ; enable interupt on done
ad.don  = 200    ; done/ready/interupt request
ad.adr  = 400    ; base of mux address (high byte)
ad.ere  = 40000 ; enable interrupt on error
ad.err  = 100000; error interrupt request
;
ad.idn  = 10000 ; data buffer identify bit
;
; define kwv csr bit offsets
;
kw.go   = 1      ; start counting immediately
kw.mod  = 2      ; base bit of count mode
kw.rat  = 10     ; base bit of rate specifier
kw.ine  = 100    ; interupt on overflow enable
kw.ovf  = 200    ; overflag flag/done
km.st1  = 400    ; maint - simulate st1
km.st2  = 1000   ; maint - simulate st2
km.osc  = 2000   ; maint - simulate oscillator single cycle
km.dio  = 4000   ; maint - disable internal oscillator
kw.ovr  = 10000 ; error overrun flag
kw.gs2  = 20000 ; enable start on st2 overflow
kw.s2e  = 40000 ; interrupt on st2 enable
kw.st2  = 100000; st2 set flag/ interrupt request
.page
;
; local storage
;
spsave: .word   0       ; saved sp for error return
chtab:  .word   0       ; storage for channel table address
chcnt:  .word   0       ; temp stoage for channel count
kwflag: .word   0       ; rate  mode words for clock
dump:   .word   0       ; place to dump junk
;
; entry point
;
getbuf::
        clr     @ adcsr
        clr     @ kwcsr
        mov     sp,spsave       ; save sp for error return on this par.
        geta    prm.3,r2        ; get address of channel table
        mov     r2,r0           ; copy addr for later
        mov     r2,chtab        ; save address again
        getp    prm.4,r3        ; get number of channels
        mov     r3,r1           ; save for later
        mov     r3,chcnt        ; save count again
;
; check all channels for validity
;
1$:     bitb     C17,(r0)+      ; check for any high bits
        beq     10$             ; continue
        sub     r2,r0           ; calc element in error
        add      10000,r0       ; indicate not a statement no
        prmerr  r0              ; force error call  traceback
10$:    dec     r1              ; check all of list
        bne     1$              ;
;
```

```
        geta    prm.1,r0        ; get address of data buffer
        getp    prm.2,r1        ; get count of samples
        mov     r0,r4           ; copy address
        ror     r4              ; get low bit
        bcc     15$             ; odd address if set
        prmerr   prm.1          ; force bad buffer address error
;
15$:    getp    prm.5,r4        ; check clock rate
        bit      C7,r4          ; check clock rate (only 1-7)
        beq     20$             ; ok if equal
        prmerr   prm.5          ; flag parameter number
;
20$:    asl     r4              ; shift
        asl     r4              ;       into
        asl     r4              ;               position
        mov     r4,kwflag       ; save into temp
        getp    prm.6,r4        ; get number of intervals
        neg     r4              ; clock counts up to zero
        mov     r4,@ kwbpr      ; load clock preset buffer
;
        bis      kw.mod*1,kwflag ; set to repeat interval
        mov      kw.go,-(sp)    ; assume no wait
        getp    prm.7,r4        ; get flags word
        bne     30$             ; good guess
        mov      kw.gs2,(sp)    ; set up for wait for st2
30$:    bis     (sp)+,kwflag    ; set into mask word
;
; assume no conversion errors
;
        putp    prm.8           ; clear error indicator
        mov      ad.cle,@ adcsr ; enable clock starts
.page
;
;       note:
;
; the following code runs at priority 7 ( non-interruptible)
; to make data acquisition as fast as possible.  to drive
 the adv11-aa
; at its fastest rate the loop below must complete within
 about
; 40 microsec nominal.  this includes 9 ms settling time for
 the amps and
; 32 ms conversion time.  at this point all parameters have
 been validated
; and set up in the following registers:
;
;       r0 - current buffer pointer (next data goes here)
;       r1 - number of samples remaining
;       r2 - pointer to channel table (next channel to be sampled)
;       r3 - channels done in this scan
;       r4 - adc csr address
;       r5 - fortran argument pointer
;
; main sampling loop...
;
        mtps     pr7            ; lock out interrupts
        mov      adcsr,r4
        mov     kwflag,@ kwcsr  ; ok, start up clock
;
bloop:  movb    (r2)+,1(r4)     ;;; set up channel
loop:   tstb    @ adcsr         ;;; wait for adc to finish
        bpl     loop            ;;; loop until done
        mov     2(r4),dump       ;; dump first conv.
        clr     2(r4)
        inc     @ adcsr          ;; then get another
```

```
looop:  tstb    @ adcsr
        bpl     looop
        mov     2(r4),(r0)+     ;;; save away data
        mov     2(r4),dump       ;; ensure bit 7 unset
        clr     2(r4)
        dec     r3              ;;; count down channels
        beq     10$             ;;; continue
        movb    (r2)+,1(r4)     ;;; set up next channel
        inc     @ adcsr         ;;; start up adc
        br      loop            ;;; go again

10$:
 if df errmrk   ;;; errmrk - flag data overruns if eq 0, mark
 data if eq 1

        .if eq errmrk-1
        bic       ad.ide,@r4    ;;; assume no error
        .endc
        tst     @r4             ;;; data overrun??
        bpl     20$             ;;; nope
        .if eq errmrk-1
        bis       ad.ide,@r4    ;;; set to mark data in error
        .endc
        dec     @prm.8(r5)      ;;; count errors
20$:
        .endc   ;;; errmrk - flag errors

        mov     chtab,r2        ;;; reset channel table address
        mov     chcnt,r3        ;;; reset channel count
        dec     r1              ;;; count down sample sets
        bne     bloop           ;;; go again
;
; all done now
;
        clr     @ adcsr         ;;; reset adc
        clr     @ kwcsr         ;;;   and clock
        mtps     pr0            ; allow interrupts
;
        return                  ; return to main program
;
        .end
```

APPENDIX E. ARCHITECTURE OF DATA STORAGE FILE

To allow the data to be viewed by the investigator using the available utility routines (pip, etc.), the data are stored as a sequence of four entries per trial. As currently implemented, the first entry is the latency to the most negative excursion within the cursors in the first channel, and the second is the magnitude of the excursion in mV. The third and fourth entries refer to the most negative excursions between the cursor sets on the second channel. The "end of file" mark is implemented as a negative latency.

The data are written into a file with "formatted" and "sequential access" attributes as a series of I5,3F10.4 records.

APPENDIX F. PROGRAMS FOR AUXILIARY ANALYSIS

At present, one auxiliary program exists that uses the data files created to perform subsequent analyses of the data.

PROGRAM: REG.FOR

PURPOSE: To determine interactions between variables

METHOD: Scatterplot of variables shown on VT55 screen and display of Pearson's r calculated for the data set

DEFENSE NUCLEAR AGENCY
ARMED FORCES RADIOBIOLOGY RESEARCH INSTITUTE
BETHESDA, MARYLAND 20814

OFFICIAL BUSINESS
PENALTY FOR PRIVATE USE, $300